UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/826,762	04/16/2004	John Harper	P3352US1 (119-0033US)	1223

61947 7590 03/25/2009
WONG, CABELLO, LUTSCH, RUTHERFORD & BRUCCULERI LLP
20333 SH 249
SUITE 600
HOUSTON, TX 77070

EXAMINER	
HSU, JONI	

ART UNIT	PAPER NUMBER
2628	

MAIL DATE	DELIVERY MODE
03/25/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/826,762
Filing Date: April 16, 2004
Appellant(s): HARPER ET AL.

William M. Hubbard

For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed December 22, 2008 appealing from the Office action mailed July 21, 2008.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,215,495	GRANTHAM	4-2001

6,995,765	BOUDIER	2-2006
6,600,840	MCCROSSIN	7-2003
6,411,301	PARIKH	6-2002
6,867,779	DOYLE	3-2005
6,801,202	NELSON	10-2004
6,919,906	HOPPE	7-2005
5,854,637	STURGES	12-1998
2002/0033844	LEVY	3-2002
6,977,661	STOKES	12-2005

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 101

1. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

2. Claim 84 is rejected under 35 U.S.C. 101 because claimed invention is directed to non-statutory subject matter. Claim 84 is directed to application program interface. One of ordinary skill in the art would define API as set of declarations of functions that operating system, library or service provides to support requests made by computer programs. Since API is set of declarations of functions, one of ordinary skill in the art would consider API to be data structure. Definition of "data structure" is "physical or logical relationship among data elements, designed to support specific data manipulation functions." Data structures per se are descriptive material

per se and are not statutory because they are not capable of causing functional change in the computer. Such claimed data structures do not define any structural and functional interrelationships between data structure and other claimed aspects of invention which permit data structure's functionality to be realized (see MPEP 2106.01). So, API is directed to non-statutory subject matter.

3. Examiner makes note Claim 1 recites "…running at least a portion of said compiled program to apply a function of said filter to said image, yielding an pixel-image result." Claim 1 is inherently tied to a machine, because machine is needed in order to run the compiled program. So, Claims 1-21 are directed to statutory subject matter. Claim 28 recites "...compiling said optimized graph…" Claim 28 is inherently tied to a machine, because a machine is needed in order to compile the optimized graph. So, Claims 28-42 are directed to statutory subject matter.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

5. Claims 28, 29, 38, 39, 43, 44, 53, 54, and 74 are rejected under 35 U.S.C. 102(b) as being anticipated by Grantham (US006215495B1).

6. As per Claim 28, Grantham teaches processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML

file. VRML file that is requested instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including Graphics State classes that includes Context, Appearance, Material, Texture, TexTransform classes (c. 28, ll. 14-16). Context class maintains graphics state for particular graphics context (c. 3, ll. 2-4). The other Graphics State classes define how result image is to be created (c. 8, ll. 38-46). So Graphics State classes indicate parameters associated with creation of result image. DrawAction class is used to draw scene (c. 7, ll. 31-33). Since VRML file that is requested includes this information about Context class, Graphics State classes, DrawAction class, VRML file that is requested requests creation of context (Context class), requests creation of result image (DrawAction class), and includes information that indicates parameters associated with creation of result image (Graphics State classes). Since 1st process requests this VRML file, 1st process is considered to request creation of context, request creation of result image, and indicate parameters associated with creation of result image. So Grantham teaches method of creating result image having 1st process requesting creation of context; 1st process requesting creation of result image; 1st process indicating parameters associated with creation of result image; 1st process requesting result image be rendered to context (c. 4, ll. 63-c. 5, ll. 23; c. 28, ll. 14-16; c. 3, ll. 2-24; c. 8, ll. 38-46; c. 7, ll. 31-33); 2nd process servicing requests of 1st process, servicing comprising steps of optimizing graph representing result image; compiling optimized graph; causing rendering of compiled optimized graph into context (c. 5, ll. 2-23; c. 4, ll. 55-58; c. 7, ll. 50-57; c. 3, ll. 2-4). Interpreter 111 modifies scene graph to suit specific graphics subsystem hardware. Interpreter enables VRML file to be adapted to run on virtually any type of machine

(c. 5, ll. 5-9). So interpreter translates VRML file to another computer language that is able to be run on graphics subsystem hardware, so interpreter is considered to perform compiling.

7. As per Claim 29, Grantham teaches creation of result image comprises editing pre-existing image (c. 11, ll. 22-25).

8. As per Claim 38, Grantham teaches 1st process is application program (112) (c. 5, ll. 2-5).

9. As per Claim 39, Grantham teaches second process comprises suite of graphics services functions (c. 3, ll. 1-16; c. 5, ll. 2-19).

10. As per Claim 43, it is similar in scope to Claim 28, except it has additional limitation that 1st process and 2nd process are running on first microprocessor, and rendering occurs on second microprocessor. Grantham teaches first process (requesting) and second process (compiling) are running on first microprocessor (108), and rendering occurs on second microprocessor (109) (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57). So Claim 43 is rejected under same rationale as Claim 28.

11. As per Claims 44, 53, and 54, these claims are similar in scope to Claims 29, 38, and 39 respectively, and therefore are rejected under the same rationale.

12. As per Claim 74, Grantham teaches rendering image, 1st process running on CPU 107 requesting creation of image; graphics services resource 109, responding to request by running 1st routine on CPU, 1st routine for optimizing graph representing image; 1st process requesting rendering of image to specified context; graphics services resource causing rendering of graph representing image, rendering occurring on GPU (c. 4, ll. 55-58, 63-67; c. 5, ll. 1-23, 61-67). Grantham recites "processor 108 of computer system 116 initiating a request which is routed by input/output (I/O) device 107 through the Internet 106 to server 101" (c. 4, ll. 63-66). So first process is running on CPU 116 requesting creation of image. Grantham teaches request results in

VRML file being transmitted to computer system 116, and VRML file instructs API 112, and API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including CompileAction class 403 that compiles specified subgraph into data structure which is more efficient for traversals (c. 7, ll. 50-57). So, ultimately API 112 responds to request by running first routine on CPU 116, first routine for optimizing graph representing the image (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57), and so API 112 is considered to be graphics services resource.

13. Thus, it reasonably appears that Grantham describes or discloses every element of Claims 28, 29, 38, 39, 43, 44, 53, 54, and 74 and therefore anticipates the claims subject.

14. Claims 75 and 78 are rejected under 35 U.S.C. 102(e) as being anticipated by Boudier (US006995765B2).

15. As per Claim 75, Boudier teaches converting 1st image representation into 2nd image representation, comprising creating 1st graph associated with 1st image representation (c. 1, ll. 29-36; c. 2, ll. 31-34; c. 3, ll. 61-63) where software routines for creating such graph execute on CPU (504) (c. 1, ll. 49-51; c. 6, ll. 34-38), determining intersection of 1st graph's global domain of definition (input scene graph) and global region of interest (bounding box) (c. 9, ll. 12-21); resolving 1st node in 1st graphic by running software routines on CPU to determine if 1st node may be combined with 2nd node (c. 9, ll. 40-53; c. 5, ll. 33-37) and create program steps for calculating and storing only portions of any intermediary image that relate to intersection (c. 9, ll. 12-21), program steps for compilation on CPU, execution on GPU (c. 5, ll. 33-37; c. 6, ll. 19-21).

16. As per Claim 78, Boudier teaches first node is root node (c. 8, ll. 10-11).

17. Thus, it reasonably appears that Boudier describes or discloses every element of Claims 75 and 78 and therefore anticipates the claim subject.

18. Claims 79-84 are rejected under 35 U.S.C. 102(e) as being anticipated by McCrossin (US006600840B1).

19. As per Claim 79, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35). Transform object 103 determines actual image vector 139, which describes format of image to be read or written (c. 6, ll. 51-53). Since this describes format of image, this is related to image objects and context objects. Image request vector is received from application and is compared to actual image vector 139 to determine what filters are needed (c. 6, ll. 53-56). Filters are accessed by transformation object from filter library. Filter library contains number of different filters available for performing various image transformations (c. 6, ll. 59-62). Pointer to filter vector in filter object 150 points to filter vector 154, which in depicted example is crop filter (c. 7, ll. 19-21). So, McCrossin teaches image processing API and API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. McCrossin uses API to call at filters. So, McCrossin teaches image processing application program interface embodied on one or more computer readable media (c. 6, ll. 25-30), having 1st group of services related to filter objects (c. 6, ll. 59-67); 2nd group of service related to image objects; 3rd group of services related to context objects (c. 5, ll. 59-60; c. 6, ll. 51-56); 4th group of services related to vector objects (c. 7, ll. 14-17).

20. As per Claim 80, McCrossin teaches 1st group of services have 1st functions to create filter objects; 2nd functions to set filter object parameters; 3rd functions to obtain filter object output (c. 2, ll. 25-32).

21. As per Claim 81, McCrossin teaches 2nd group of services have 1st function to create image objects (c. 6, ll. 51-56); 2nd functions to render an image object to context object (c. 9, ll. 7-18).

22. As per Claim 82, McCrossin teaches 3rd group of services have 1st functions to create context objects (c. 5, ll. 46-67).

23. As per Claim 83, McCrossin teaches 4th group of services comprise 1st functions to create vector objects (c. 7, ll. 15-32).

24. As per Claim 84, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35). Filters are accessed by transformation object from filter library 143. Filter library contains number of different filters available for performing various image transformations (c. 6, ll. 59-62). Transform object adds filter to filter stack to create new filter object (c. 6, ll. 51-58; c. 8, ll. 25-37). Since McCrossin teach creating new file object for filter stack (c. 8, ll. 25-37), this is considered to be creating filter object. According to Applicant's disclosure, image, more commonly, may be created from another image by applying filter to existing image ([0057], p. 12). So, McCrossin teaches API functions to create image objects; API functions to create filter objects. Transform object 103 determines actual image vector 139, which describes format of image to be read or written (c. 6, ll. 51-53). According to Applicant's disclosure, creating context is derived from tools that allow definition of object in memory ([0055], p. 11). Since actual image vector describes format of image to be read or written, this allows definition of object in memory, so is considered to create context objects, so McCrossin teaches API functions to create context objects. Pointer to filter environment in filter object 150 points to memory 156, which contains information which may be used by filter vector 154 in

converting or transforming image data (c. 7, ll. 21-24), and so McCrossin does teach API functions to set filter object parameters. Filters are employed by transformation object 103 to provide necessary transformation of image data to return image data in form as specified in image request vector 127 (c. 6, ll. 66-c. 7, ll. 2), and so McCrossin teaches API functions to obtain filter object output; API functions to convert image objects to context objects. So, McCrossin teaches application program interface for facilitating image processing (c. 6, ll. 25-30), API having functions to create image objects; create context objects (c. 6, ll. 51-53); create filter objects; set filter object parameters; obtain filter object output (c. 2, ll. 25-32); convert image objects into context objects (c. 6, ll. 51-53).

25. Thus, it reasonably appears that McCrossin describes or discloses every element of Claims 79-84 and therefore anticipates the claims subject.

Claim Rejections - 35 USC § 103

26. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

27. The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

28. Claims 1, 2, 8, 11, 12, 14-20, 36, 37, 40, 41, 51, 52, 55, 56, 58, 61-63, 66-73, and 85 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1).

29. As per Claim 1, Grantham teaches processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML file. VRML file instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. Interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Hence, interpreter 111 enables VRML file to be adapted to run on graphics subsystem hardware (c. 4, ll. 63-c. 5, ll. 11). API is structured as collection of class hierarchies. Nodes and graphics states are assembled into cohesive scene graph (c. 5, ll. 18-23). Graphics State classes include Texture class 504 that defines how image is filtered (c. 28, ll. 10-16; c. 11, ll. 22-25). Outside Scene Graph classes include CompileAction class 403 that compiles specified subgraph into data structure which is more efficient for traversals (c. 28, ll. 25-27, 29-30; c. 7, ll. 50-57). So filter is part of scene graph (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25) that is run on graphics subsystem hardware (c. 5, ll. 5-11). Before scene graph is run on graphics subsystem hardware (c. 5, ll. 5-11), interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Interpreter 111 enables VRML file to be adapted to run on virtually any type of machine (c. 5, ll. 5-9). So interpreter 111 translates VRML file to another computer language that is able to be run on graphics subsystem hardware 109, and so interpreter 111 is considered to perform compiling, and compiled program includes filter. So VRML file requested by processor 108 (c. 4, ll. 63-c. 5, ll. 11) includes filter (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25), and processor 108 requests

filter from compiling process since program needs to be compiled before it is run on graphics system hardware. So Grantham teaches method of editing initial image, having steps of 1st process requesting (c. 4, ll. 63-c. 5, ll. 11) filter from 2nd process; related filter and initial image comprising program, 2nd process compiling program, yielding compiled program; running at least portion of compiled program to apply function of filter to image (c. 5, ll. 5-11, 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25; c. 7, ll. 50-57), yielding pixel-image result (c. 5, ll. 9-11).

However, Grantham does not teach 1st process defines relationship between filter and initial image. But, McCrossin teaches editing initial image, 1st process requesting filter from 2nd process; 1st process defining relationship between filter and initial image (c. 6, ll. 46-c. 7, ll. 9).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham so first process defines relationship between filter and initial image because McCrossin suggests reducing amount of processing required to convert image data from original or input format into desired output format (c. 1, ll. 54-57; c. 2, ll. 3-24).

30. As per Claim 2, Grantham teaches optimizing the program (c. 4, ll. 55-58).

31. As per Claim 8, Grantham teaches compiled program has component for 1st (108, Fig. 1) and 2nd microprocessors (109) (c. 4, ll. 63-c. 5, ll. 11; c. 7, ll. 50-57).

32. As per Claim 11, Grantham teaches 1st microprocessor is CPU (108, Fig. 1) and 2nd microprocessor is GPU (109) (c. 4, ll. 63-c. 5, ll. 11).

33. As per Claim 12, Grantham teaches 1st and 2nd microprocessors are both CPUs (108, 102, Fig. 1; c. 4, ll. 63-c. 5, ll. 11).

34. As per Claim 14, Grantham teaches initial image is only color (c. 12, ll. 45-52).

35. As per Claim 15, Grantham teaches 1st process is application program (112) (c. 5, ll. 2-5).

36. As per Claim 16, Grantham teaches 2nd process has suite of graphics services functions (c. 3, ll. 1-16; c. 5, ll. 2-19).

37. As per Claim 17, Grantham does not explicitly teach operating system comprises second process. However, McCrossin teaches this limitation (c. 4, ll. 2-5).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify device of Grantham so operating system comprises second process because McCrossin suggests operating system is needed in order for processor to function (c. 4, ll. 2-5).

38. As per Claim 18, Grantham teaches VRML is high level programming language (c. 1, ll. 63-67). Processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML file. VRML file instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. Interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Hence, interpreter 111 enables VRML file to be adapted to run on graphics subsystem hardware (c. 4, ll. 63-c. 5, ll. 11). API is structured as collection of class hierarchies. Nodes and graphics states are assembled into cohesive scene graph (c. 5, ll. 18-23). Graphics State classes include Texture class 504 that defines how image is filtered (c. 28, ll. 10-16; c. 11, ll. 22-25). So high level programming language VRML (c. 1, ll. 63-67) is interpreted by interpreter 111 into lower level programming language so that it is adapted to run on graphics subsystem hardware (c. 5, ll. 5-11), this lower level programming language includes filter (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25). So 1st process requests filter by requesting VRML file (c. 4, ll. 63-67), which is high level programming language (c. 1, ll. 63-67), 2nd process responds by interpreting high level programming language into lower level programming

language (c. 5, ll. 5-11) including filter so that filtering is adapted to run on graphics subsystem hardware (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25). So 1st process requests high-level filter (c. 4, ll. 63-67; c. 1, ll. 63-67) and 2nd process responds with object representing lower-level filter (c. 5, ll. 5-11, 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25).

39. As per Claim 19, Grantham shows processor 108, interpreter 111, and API 112 are part of computer system 116 in Fig. 1. So first process and second process run on CPU (108, 111, 112) and compiled program runs on GPU (109) (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57).

40. As per Claim 20, it is similar in scope to Claim 19, so is rejected under same rationale.

41. As per Claim 36, Grantham teaches first process requests output of graph programmatically assembled by first process (c. 4, ll. 63-c. 5, ll. 23).

However, Grantham does not explicitly teach graph has one or more pre-defined filters. But, McCrossin teaches this (c. 6, ll. 59-67). This would be obvious for reasons for Claim 1.

42. As per Claim 37, Grantham teaches first process programmatically assembled graph in cooperation with second process (compiling) (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57).

43. As per Claims 40-41, these claims are each similar in scope to Claim 17, and so are rejected under same rationale. As per Claims 51, 52, 55, and 56, these claims are similar in scope to Claims 36, 37, 40, and 41 respectively, and so are rejected under the same rationale.

44. As per Claim 58, Grantham teaches interpreter 111 enables VRML file to be adapted to run on graphics system hardware 109 (c. 5, ll. 5-9). VRML is high level programming language (c. 1, ll. 63-67). So Grantham teaches providing high level interface (111) to graphics processing resource (109) (c. 5, ll. 5-9; c. 1, ll. 63-67) having 1st process requesting performance of task through 1 or more function calls serviced by graphics processing resource (c. 4, ll. 63-c. 5, ll.

23), function calls selected from 1 or more of following options, (i) creating of image (Graphics State classes) (c. 28, ll. 14-16; c. 8, ll. 37-46; c. 7, ll. 31-33), (iv) asking for output of filter or another filter (504; c. 11, ll. 22-25), (v) creating context (302); (vi) rendering image to context or another context; request having object associated therewith, object comprising one of following: context (c. 5, ll. 61-67), image (c. 28, ll. 14-16; c. 8, ll. 37-46; c. 7, ll. 31-33), filter (c. 11, ll. 22-25), or vector (c. 26, ll. 20-22); graphics processing resource servicing request (c. 5, ll. 2-23).

However, Grantham does not explicitly teach request defines relationship between at least one of functions and one of objects. However, McCrossin teaches this limitation (c. 6, ll. 46-c. 7, ll. 9). This would be obvious for the same reasons given in the rejection for Claim 1.

45. As per Claim 61, Grantham teaches creating graph representing image (c. 2, ll. 52-58).

46. As per Claim 62, Grantham teaches optimizing graph (c. 4, ll. 55-59; c. 5, ll. 2-11).

47. As per Claims 63, 66, and 67, these claims are similar in scope to Claims 58, 61, and 62 respectively, and so are rejected under same rationale.

48. As per Claim 68, Grantham does not explicitly teach function calls include option of creating filter. But, McCrossin teaches this (c. 6, ll. 59-67). This is for reasons for Claim 1.

49. As per Claim 69, Grantham does not explicitly teach function calls include option of setting arguments associated with filter. However, McCrossin teaches this limitation (c. 7, ll. 1-7; c. 6, ll. 9-21). This would be obvious for the same reasons given in the rejection for Claim 1.

50. As per Claim 70, it is similar in scope to Claim 69, so is rejected under same rationale.

51. As per Claim 71, Grantham does not teach another object associated with request has filter. But, McCrossin teaches this (c. 6, ll. 59-67). This is obvious for reasons for Claim 1.

52. As per Claim 72, Grantham does not teach another object associated with request is vector. But, McCrossin teaches this (c. 7, ll. 14-17). This is obvious for reasons for Claim 1.

53. As per Claim 73, it is similar in scope to Claim 72, so is rejected under same rationale.

54. As per Claim 85, Grantham teaches computer executable instructions for performing method recited in Claim 1 (c. 4, ll. 63-c.5, ll. 23).

55. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1) in view of Levy (US 20020033844A1).

Grantham and McCrossin are relied upon for teachings for Claim 2.

But, Grantham and McCrossin do not teach optimizing includes step of using cache look-up to see if pixel-image result is already in cache. But, Levy teaches this [0184, 0021, 0038].

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham, McCrossin so optimizing includes using cache look-up to see if pixel-image result is already in cache because Levy teaches avoiding repeated read operations on same content [0184].

56. Claims 4-6 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1) in view of Boudier (US006995765B2).

57. As per Claim 4, Grantham and McCrossin are relied on for teachings relative Claim 2.

But, Grantham and McCrossin do not teach optimizing includes step of calculating intersection, intersection representing area where pixel-image result is both defined and in result region requested of 2nd process. But, Boudier teaches step of optimizing includes step of calculating intersection, intersection representing area where pixel-image result is both defined (input scene graph) and in result region requested of 2nd process (bounding box) (c. 9, ll. 12-21).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham, McCrossin so optimizing includes calculating intersection, intersection representing area where pixel-image result is both defined and in result region requested of 2nd process because Boudier teaches removing unnecessary regions, which reduces traversal time (c. 9, ll. 12-21).

58. As per Claim 5, Grantham does not teach step of using calculated intersection to limit number of pixels that require calculation during running of compiled program. However, Boudier teaches step of using calculated intersection (intersection of input scene graph and bounding box) to limit number of pixels that require calculation during running of compiled program (c. 9, ll. 12-21; c. 8, ll. 52-53). This would be obvious for reasons for Claim 4.

59. As per Claim 6, Grantham doesn't teach using calculated intersection to limit amount of memory necessary for storing calculated images. Boudier teaches memory usage is increased during actual calculating of intersection due to creation of bounding volumes. But, after creation of bounding boxes, after the intersection is calculated, the unnecessary bounding boxes are removed. Bounding box is used for view frustum culling algorithm, so data outside of bounding box is culled out, so only data inside bounding box needs to be stored, so calculated intersection (intersection of input scene graph and bounding box) is used to limit amount of memory necessary for storing calculated images (c. 9, ll. 12-21). This is obvious for reasons for Claim 4.

60. As per Claim 21, it is similar in scope to Claim 19, so is rejected under same rationale.

61. Claims 7, 9, 60, and 65 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1) in view of Parikh (US006411301B1).

62. As per Claim 7, Grantham and McCrossin are relied on for teachings relative to Claim 1.

However, Grantham and McCrossin do not teach compiled program is for single microprocessor. However, Parikh teaches single microprocessor that emulates graphics processing (c. 26, ll. 29-62). So, Grantham can be modified so only single microprocessor is used instead of using separate graphics processor (c. 5, ll. 2-23) so compiled program of Grantham (c. 7, ll. 50-57) is for single microprocessor, as suggested by Parikh.

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham and McCrossin so compiled program is for single microprocessor because Parikh suggests advantage of still being able to perform graphics processing even when graphics processor for which software is written for is not available (c. 26, ll. 29-62).

63. As per Claim 9, Grantham does not teach single microprocessor is CPU. However, Parikh teaches this limitation (c. 26, ll. 29-62). This would be obvious for reasons given for Claim 7.

64. As to Claim 60, Grantham doesn't teach request is serviced using emulator to run GPU program on CPU. But, Parikh teaches this (c. 26, ll. 29-62). This is obvious for reasons for Claim 7.

65. As per Claim 65, it is similar in scope to Claim 60, so is rejected under same rationale.

66. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1), McCrossin (US006600840B1), and Parikh (US006411301B1) in view of Doyle (US006867779B1).

Grantham, McCrossin, and Parikh are relied on for teachings relative to Claim 7.

However, Grantham, McCrossin, and Parikh do not teach single microprocessor is programmable GPU. However, Doyle teaches dividing up rendering between CPU and GPU based on progress of one device of two devices (c. 1, ll. 20-26; c. 2, ll. 1-3). So, if CPU is busy, rendering is only performed in GPU. So, single microprocessor is programmable GPU.

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham, McCrossin, and Parikh so single microprocessor is programmable GPU because Doyle teaches using device that would most quickly process command to process command (c. 1, ll. 20-26).

67. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1) in view of Nelson (US006801202B2).

Grantham and McCrossin are relied on for teachings for Claim 8.

But, Grantham and McCrossin do not teach 1st and 2nd microprocessors are both GPUs. But, Nelson teaches API to generate commands and graphics data, and these commands are processed by 1st and second microprocessors that are both GPUs (c. 8, ll. 32-42, 67; c. 9, ll. 1-3).

It would have been obvious to one of ordinary skill in the art to modify devices of Grantham and McCrossin so 1st and 2nd microprocessors are both GPUs because Nelson suggests advantage of increased graphics performance (c. 3, ll. 47-c. 63).

68. Claims 22, 23, and 25-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1) in view of Hoppe (US006919906B2).

69. As per Claim 22, Grantham teaches editing initial image, 1st microprocessor (108, 111, 112, Fig. 1) running 1st process and 2nd process (compiling) for servicing requests from first process (c. 4, ll. 63-c. 5, ll. 11; c. 7, ll. 50-57). API is structured as collection of class hierarchies (c. 5, ll. 16-23) that include Texture class 504 that defines how image is filtered (c. 11, ll. 22-25). So filter is in API (c. 5, ll. 16-23; c. 11, ll. 22-25), and API 112 is stored in memory 110, as shown in Fig. 1. So memory 110 is for storing filter, 2nd memory for storing data structure comprising information used in 1st process (c. 5, ll. 16-23; c. 11, ll. 22-25; Fig. 1) (1st and 2nd memories are same, as recited in Claim 23); 2nd microprocessor 109 for running program created using data structure; outputting pixel image resulting from running program (c. 5, ll. 2-11).

But, Grantham doesn't teach data structure has relationship between initial image and filter. But, McCrossin teaches data structure 103 has relationship between initial image and specific filter (*image request vector 127 is received from application 101 and is compared to actual image vector to determine what filters are needed*, c. 6, ll. 46-c. 7, ll. 9; *application calls the transformation object using API call*, c. 7, ll. 33-35). Fig. 6 shows transform object 103 (data structure) comprises image request vector 127 that is compared to actual image vector to determine what filters are needed (c. 6, ll. 46-c. 7, ll. 9). Since transform object 103 compares actual image vector to determine what filters are needed, this means that transform object 103 determines from image what filters are needed, and so transform object 103 (data structure) comprises relationship between image and filter. This is obvious for reasons for Claim 1.

But, Grantham, McCrossin don't teach 3rd memory for storing pixel image resulting from running program. But, Hoppe teaches API (c. 2, ll. 20-26) for filtering (c. 1, ll. 34-36), 3rd memory 208 for storing pixel image resulting from running program (c. 4, ll. 36-37; c. 9, ll. 33-36; c. 1, ll. 55-58).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham and McCrossin to include 3rd memory for storing pixel image resulting from running program because Hoppe suggests it is well-known in the art to have frame buffer in video memory to store rendered image so that rendered image can be output from video memory to display (c. 4, ll. 36-37; c. 9, ll. 33-36).

70. As per Claim 23, Grantham teaches 1st and 2nd memories are the same (Fig. 1; c. 4, ll. 18-21; c. 5, ll. 2-23).

71. As per Claim 25, Grantham teaches 1st and 2nd memories are in system memory (110, Fig. 1; c. 4, ll. 18-21; c. 5, ll. 2-23)

However, Grantham does not teach third memory is in video memory. However, Hoppe teaches this limitation (208; c. 4, ll. 36-37). This would be obvious for reasons for Claim 22.

72. As per Claim 26, Grantham teaches first microprocessor (108, Fig. 1) processes data structure to produce program for use on second microprocessor (109) (c. 4, ll. 63-c. 5, ll. 11).

73. As per Claim 27, Grantham teaches second microprocessor (109, Fig. 1), under control of program causes pixel image to be output (c. 5, ll. 2-11).

But, Grantham doesn't teach image is stored in 3rd memory. But, Hoppe teaches 2nd processor 206, under control of program causes pixel image to be stored in 3rd memory 208 (c. 4, ll. 36-44; c. 9, ll. 33-36; c. 1, ll. 55-58; c. 2, ll. 20-26). This is obvious for reasons for Claim 22.

74. Claim 24 is rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1), McCrossin (US006600840B1), and Hoppe (US006919906B2) in view of Sturges (US005854637A).

Grantham and McCrossin are relied on for teachings relative to Claim 22. Grantham teaches first and second memories are system memory (Fig. 1; c. 4, ll. 18-21; c. 5, ll. 2-23). Hoppe teaches third memory is frame buffer (208; c. 4, ll. 36-37), as discussed for Claim 22.

But, Grantham, McCrossin do not teach 1st, 2nd, 3rd memories are the same. But, Sturges teaches using shared memory for both system memory and frame buffer (c. 3, ll. 7-15). So combination of Grantham, McCrossin can be modified so 1st and 2nd memories in system memory of Grantham, frame buffer of Hoppe are in the same memory as suggested by Sturges.

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham and McCrossin so 1st, 2nd and 3rd memories are the same because Sturges teaches using unused portions of frame buffer as system memory (c. 1, ll. 35-41).

75. Claims 30-32 and 45-47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) in view of Boudier (US006995765B2).

76. As per Claim 30, it is similar in scope to Claim 4, and so is rejected under same rationale.

77. As per Claim 31, Grantham does not teach step of optimizing graph representing first image comprises step of analyzing adjacent nodes in graph for purpose of attempting to consolidate nodes. However, Boudier teaches this (c. 9, ll. 40-53; c. 8, ll. 4-10).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham so optimizing graph representing first image comprises analyzing adjacent nodes in graph for purpose of attempting to consolidate nodes because Boudier teaches this reduces the number of nodes, so reduces memory usage, and drawing time is reduced because there will be fewer function calls (c. 8, ll. 20-24; c. 9, ll. 40-53).

78. As per Claim 32, it is similar in scope to Claim 31, so is rejected under same rationale. As per Claims 45-47, these claims are similar in scope to Claims 30-32 respectively, so are rejected under same rationale.

79. Claims 33 and 48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and Boudier (US006995765B2) in view of Levy (US 20020033844A1).

Grantham and Boudier are relied upon for teachings as discussed relative to Claim 31.

But, Grantham doesn't teach analyzing adjacent nodes and storing result of such analysis in memory. But, Boudier teaches this (c. 9, ll. 40-53; c. 8, ll. 20-21). This would be obvious for reasons for Claim 31.

But, Grantham and Boudier do not teach checking cache to determine if result of such analysis is available in memory. But, Levy teaches step of checking cache to determine if pixel-image result is available in memory [0184, 0021, 0038]. This is obvious for reasons for Claim 3.

80. Claims 34, 35, 49, and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) in view of Levy (US 20020033844A1).

81. As per Claim 34, Grantham is relied on for teachings for Claim 28. Grantham teaches step of optimizing graph and storing optimized graph in memory (c. 4, ll. 55-58; c. 2, ll. 54-67).

However, Grantham does not teach step of checking cache to determine if graph has already been optimized. But, Levy teaches step of checking cache to determine if pixel-image result is already in memory [0184, 0021, 0038]. This would be obvious for reasons for Claim 3.

82. As per Claim 35, Grantham teaches step of optimizing graph and outputting result of rendering graph (c. 4, ll. 55-58; c. 5, ll. 2-23).

But, Grantham does not teach step of checking cache to determine if result of rendering graph is available in memory. But, Levy teaches step of checking cache to determine if pixel-image result is available in memory [0184, 0021, 0038]. This is obvious for reasons for Claim 3.

83. As per Claims 49 and 50, these claims are similar in scope to Claims 34 and 35 respectively, and so are rejected under same rationale.

84. Claims 42 and 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) in view of Stokes (US006977661B1).

Grantham is relied on for teachings for Claim 28. Grantham teaches 2nd process inserts nodes into graph (c. 5, ll. 2-23) for performing certain functions (c. 4, ll. 23-49).

However, Grantham does not teach functions include functions for converting original color scheme to operating color scheme and for converting operating color scheme back to original color scheme. However, Stokes teaches this limitation (c. 5, ll. 63-67).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham so 2nd process inserts nodes into graph for converting original color scheme to operating color scheme and for converting operating color scheme back to original color scheme because Stokes suggests this is needed because color data generated by image-capturing device are generally in device-specific color space that is different from color space or spaces used by image-processing application for image editing (c. 1, ll. 40-44).

85. Claims 59 and 64 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grantham (US006215495B1) and McCrossin (US006600840B1) in view of Doyle (US00687779B1).

Grantham and McCrossin are relied on for teachings discussed relative to Claim 58.

However, Grantham and McCrossin do not teach request is serviced using GPU and CPU. However, Doyle teaches request is serviced using GPU and CPU (c. 2, ll. 1-3).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham and McCrossin so request is serviced using GPU and CPU because Doyle teaches this accelerates speed by which 3-D image can be rendered (c. 1, ll. 7-10).

86. Claims 76-77 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boudier (US006995765B2) in view of Levy (US 20020033844A1).

87. As per Claim 76, Boudier is relied on for teachings for Claim 75. Boudier teaches determining if 1st node may be combined with 2nd node, result is stored in memory (c. 9, ll. 40-53; c. 8, ll. 20-21).

But, Boudier does not teach step of checking cache to determine if there is result is already in memory regarding such determination regarding combining nodes. But, Levy teaches step of checking cache to determine if pixel-image result is already in memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

88. As per Claim 77, Boudier teaches step of resolving first node and storing resolution of first node in memory (c. 9, ll. 40-53; c. 8, ll. 20-21).

However, Boudier does not teach step of resolving first node comprises step of checking cache to determine if there is result in memory regarding resolution of first node. However, Levy teaches step of checking cache to determine if pixel-image result is already in memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

(10) Response to Argument

1. Appellant's arguments, see p. 10, filed December 22, 2008, with respect to the 35 U.S.C. 101 rejections of Claims 79-83 and 85 have been fully considered and are persuasive. The 35 U.S.C. 101 rejections of Claims 79-83 and 85 have been withdrawn.

2. Appellant argues that Claims 79-83 and 85 are directed to statutory subject matter (p. 10).

In reply, Examiner agrees. Appellant's disclosure describes that software running on the CPU are stored in memory ([0009], p. 3), and "Memory is typically any of a variety of types of dynamic random access memory, but my also, in alternative configurations be static RAM, magnetic memory, optical memory or any other suitable storage medium" ([0032], p. 5-6). So, the computer readable medium recited in Claims 79-83 and 85 is considered to be the memory as described in Appellant's disclosure, and so Claims 79-83 and 85 are directed to statutory subject matter, and therefore the 35 U.S.C. 101 rejections of Claims 79-83 and 85 have been withdrawn.

3. Appellant's arguments filed December 22, 2008, with respect to 35 U.S.C. 101 rejection of Claim 84 and 35 U.S.C. 102 and 35 U.S.C. 103 rejections have been fully considered but they are not persuasive.

4. As per Claim 84, Appellant argues that the Examiner presented differing definitions for "data structures" and "APIs" but contradicts those definitions by incorrectly concluding that "one of ordinary skill in the art would consider API to be a data structure (p. 10).

In reply, Examiner points out that Examiner stated that API is a set of declarations of functions. Examiner stated data structure is physical or logical relationship among data elements, designed to support specific data manipulation functions, and so data structure is also a set of declarations of functions. Therefore, API and a data structure are both a set of declarations of functions, and so one of ordinary skill in the art would consider API to be a data structure.

5. Appellant argues that Claim 84 includes a concrete and tangible step of processing an image and cannot be classified as "nonfunctional" (p. 11).

In reply, Examiner points out that API is a data structure, as discussed above. Data structures per se are descriptive material per se and are not statutory because they are not capable of causing functional change in the computer. Since Claim 84 recites an API per se and does not recite a computer, **the API per se of Claim 84 does not cause functional change in the computer since no computer is recited in Claim 84**. Such claimed data structures do not define any structural and functional interrelationships between data structure and other claimed aspects of invention which permit data structure's functionality to be realized (see MPEP 2106.01). So, API is directed to non-statutory subject matter.

6. As per Claim 28, Appellant argues that the "CompileAction" of Grantham (US006215495B1) is not "compilation" in its normal defined sense or as used by assignee, because Grantham's "CompileAction" does not yield executables (p. 12).

In reply, Examiner points out that definition of compiler is computer program that translates text written in computer language into another computer language. Grantham teaches interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Interpreter 111 enables VRML file to be adapted to run on virtually any type of machine (c. 5, ll. 5-9). So interpreter 111 translates VRML file to another computer language that is able to be run on graphics subsystem hardware 109, and so interpreter 111 is considered to perform compiling.

7. As per Claim 28, Appellant argues that the local processor (i.e., 1st process) of Grantham does not "indicate parameters associated with creation of result image." The local processor is merely making an internet request of a web server and the web server provides the appropriate VRML file. The cited "first process" in Grantham merely asks for the VMRL file without any reference to or cognizance of the claimed "creation of context", "creation of result image", or "parameters associated with result image" (p. 14). As per Claim 43, the internet server of Grantham performs the function of creating the VRML file (p. 15).

Examiner points out Grantham teaches that VRML file that is requested instructs API 112, and API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including Graphics State classes that includes Context, Appearance, Material, Texture, and TexTransform classes (c. 28, ll. 14-16). Context class maintains graphics state for particular graphics context (c. 3, ll. 2-4). The other Graphics State classes define how result image is to be created (c. 8, ll. 38-46). So Graphics State classes indicate parameters associated with creation of result image. DrawAction class is used to draw scene (c. 7, ll. 31-33). Since VRML file that is requested includes this information about Context class, Graphics State classes, and DrawAction class, VRML file that is requested requests creation of context (Context class), requests creation of result image (DrawAction class), and includes information that indicates parameters associated with creation of result image (Graphics State classes). Since 1st process requests this VRML file, 1st process is considered to request creation of context, request creation of result image, and indicate parameters associated with creation of result image.

8. As per Claim 74, Appellant argues that the 1st process of Grantham is a web browser and is not "requesting creation of an image." The web browser is "initiating a request which is

routed by input/output (I/O) device 107 through the internet 106 to server 101." A web browser requesting information from a server is not the same as "a first process running on a CPU requesting creation of an image." Neither the web browser nor Grantham's server has "a graphics services resource, responding to said request" (p. 16).

In reply, Grantham recites "**processor 108 of computer system 116** initiating a request which is routed by input/output (I/O) device 107 through the Internet 106 to server 101" (c. 4, ll. 63-66). So first process is running on CPU 116 requesting creation of image. Grantham teaches request results in VRML file being transmitted to computer system 116, and VRML file instructs API 112, and API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including CompileAction class 403 that compiles specified subgraph into data structure which is more efficient for traversals (c. 7, ll. 50-57). So, ultimately API 112 responds to request by running first routine on CPU 116, first routine for optimizing graph representing the image (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57), and so API 112 is considered to be graphics services resource.

9. As per Claim 75, Appellant argues that the Assignee's graph's global domain of definition is not anticipated by Boudier's (US006995765B2) input scene graph. The instant specification states "practical way to think about domain of definition is as a representation of all places in an image that are explicitly defined and not transparent." The scene graph of Boudier is a model with the nodes representing features of a scene and edges representing associations between the connected components. The Assignee's global region of interest is not equivalent to the bounding box of Boudier. The instant specification explains that a region of interest "is the portion of the input image that is necessary to compute the given DOD" (p. 17). Claim 75 includes a limitation of "determining an intersection of the first graph's global domain of

definition and a global region of interest." Because Boudier does not disclose either a DOD or ROI as defined by Assignee, it is impossible for Boudier to anticipate the claim (p. 18).

In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., domain of definition is representation of all places in image that are explicitly defined and not transparent, region of interest is portion of input image that is necessary to compute given DOD) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993). Since Claim 75 does not recite domain of definition is representation of all places in image that are explicitly defined and not transparent, and region of interest is portion of input image that is necessary to compute given DOD, Boudier does not necessarily have to expressly teach these limitations in order to anticipate the claim. Therefore, the input scene graph of Boudier is considered to be DOD, and the bounding box of Boudier is considered to be ROI, and so Boudier teaches DOD and ROI **as recited in Claim 75**.

10. As per Claim 79, Appellant argues API discussed by McCrossin (US006600840B1) does not disclose image processing API. Rather, McCrossin's only APIs are "read" and "write" APIs. Examiner is incorrect in inferring that McCrossin teaches specific bundle of API services as claimed by merely stating an API call or procedure call is used. McCrossin does not teach APIs to filter object, APIs to image objects, APIs to context objects and APIs to vector objects (p. 19).

In reply, Examiner points out McCrossin describes "an application calls the transformation object using a procedure call or API call" (c. 7, ll. 33-34), and "transform object 103 determines an actual image vector 139...image request vector 127...is compared to actual image vector 139 to determine what filters are needed" (c. 6, ll. 51-56). So, McCrossin does teach APIs to filter objects and other objects, and does not only comprise of read and write APIs.

11. Appellant argues simply describing format of image as taught by McCrossin does not encompass requirements of image objects and context objects. Correct quotation from Assignee's specification is "the ability to create a context is derived from tools that allow definition of an object in memory." Examiner seems to have incorrectly interpreted the partial quotation to believe that any definition of an object in memory is creating a context (p. 20).

In reply, Examiner points out according to Appellant's disclosure, context is space such as defined place in memory in which result of filtering operation resides (p. 9, [0045]). McCrossin teaches buffer 107 in which result of filtering operation resides (c. 5, ll. 59-60; c. 6, ll. 51-56), and so teaches context creation.

12. Appellant argues that such a bundle of API services as claimed is entirely absent from McCrossin at least because a major purpose of McCrossin is to isolate the application layer from the task of doing a file format conversion (p. 20).

In reply, Examiner points out that major purpose of McCrossin is not really about isolating application layer from task of doing file format conversion, but actually the major purpose is to use plurality of filters for performing file format conversion, and comparing parameter values to determine whether or not filter is needed for file format conversion to efficiently transform format without performing any unnecessary work (c. 11, ll. 23-36; c. 1, ll. 50-c. 2, ll. 2). As discussed above, McCrossin does teach API services related to filter objects.

13. As per Claim 84, Appellant argues that McCrossin does not create a new filter object but creates a "new file object" and filters are only pushed and popped from a filter stack (p. 22).

In reply, Claim 84 recites "create filter **objects**". Since McCrossin teach creating new file object for filter stack (c. 8, ll. 25-37), this is considered to be creating filter object.

14. As per Claim 1, Appellant argues VRML file of Grantham is not graphics item and is much higher level collection of information than graphics filter. Because VRML file is high level file and can be interpreted into something applicable to any type of machine, there is no reason to even infer that the VRML file will contain the claimed graphics processing filter (p. 24).

In reply, the Examiner points out that Grantham teaches that graphics stages are assembled into cohesive scene graphics (c. 5, ll. 18-23). Graphics Stage classes include Texture class 504 that defines how image is filtered (c. 28, ll. 10-16; c. 11, ll. 22-25). So filter is part of scene graph (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25).

15. Appellant argues that McCrossin does not teach multiple processes (p. 24).

In reply, Examiner points out Grantham is used to teach multiple processes and specific division of work between multiple processes. McCrossin is merely used to teach 1st process defines relationship between filter and initial image.

16. As per Claim 1, Appellant argues that Grantham does not disclose a defined relationship (p. 24). As per Claim 22, McCrossin's transform object 103 does not teach a data structure comprising a relationship between an image and a filter (p. 27).

In reply, Examiner points out Fig. 6 of McCrossin shows transform object 103 (data structure) comprises image request vector 127 that is compared to actual image vector to determine what filters are needed (c. 6, ll. 46-c. 7, ll. 9). Since transform object 103 compares actual image vector to determine what filters are needed, this means that transform object 103 determines from image what filters are needed, and so transform object 103 (data structure) comprises relationship between image and filter.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Joni Hsu

/Joni Hsu/

Examiner, Art Unit 2628

Conferees:

Kee Tung

/Kee M Tung/

Supervisory Patent Examiner, Art Unit 2628

Xiao Wu

/XIAO M. WU/

Supervisory Patent Examiner, Art Unit 2628